

11-24-00

A

JC872 U.S. PTO  
11/22/00

Jc714 U.S. PTO  
09/22/00  
11/22/00

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Inventorship ..... Evans et al.  
Applicant ..... Microsoft Corporation  
Attorney's Docket No. .... MS1-688US  
Title: Improved DVD Navigator and Application Programming Interfaces (APIs)

**TRANSMITTAL LETTER AND CERTIFICATE OF MAILING**

To: Commissioner of Patents and Trademarks  
Washington, D.C. 20231  
From: Thomas A. Jolly (509) 324-9256  
Lee & Hayes, PLLC  
421 W. Riverside Avenue, Suite 500  
Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Transmittal Letter with Certificate of Mailing included.
2. PTO Return Postcard Receipt
3. Check in the Amount of \$750.00
4. Fee Transmittal
5. New patent application (title page plus 41 pages, including claims 1-12 & Abstract)
6. Executed Declaration
7. 7 sheets of formal drawings (Figs. 1-14)
8. Assignment w/Recordation Cover Sheet

Large Entity Status  Small Entity Status

The Commissioner is hereby authorized to charge payment of fees or credit overpayments to Deposit Account No. 12-0769 in connection with any patent application filing fees under 37 CFR 1.16, and any processing fees under 37 CFR 1.17.

Date: 11-22-2000 By: Thomas A. Jolly  
Thomas A. Jolly  
Reg. No. 39,241

**CERTIFICATE OF MAILING**

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EL685270489

Date: 11-22-00 By: Lori A. Vierra  
Lori A. Vierra

EL685270489

PTO/SB/17 (11-00)  
 Approved for use through 10/31/2002. OMB 0651-0032  
 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

|                                                                                                                                                                                 |                          |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------|
| <h1 style="margin: 0;">FEE TRANSMITTAL</h1> <h2 style="margin: 0;">for FY 2001</h2> <p style="font-size: small; margin: 5px 0;">Patent fees are subject to annual revision.</p> | <b>Complete if Known</b> |             |
|                                                                                                                                                                                 | Application Number       |             |
|                                                                                                                                                                                 | Filing Date              |             |
|                                                                                                                                                                                 | First Named Inventor     | Evans       |
|                                                                                                                                                                                 | Examiner Name            |             |
|                                                                                                                                                                                 | Group Art Unit           |             |
| Attorney Docket No.                                                                                                                                                             | MSI-688US                |             |
| TOTAL AMOUNT OF PAYMENT                                                                                                                                                         |                          | (\$) 750.00 |

| METHOD OF PAYMENT                                                                                                                                                                                                                     |                                                                                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. <input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to:<br>Deposit Account Number: <u>12-0769</u><br>Deposit Account Name: <u>Lee &amp; Hayes, PLLC</u> | <input checked="" type="checkbox"/> Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17<br><input type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27 |
| 2. <input checked="" type="checkbox"/> Payment Enclosed:<br><input checked="" type="checkbox"/> Check <input type="checkbox"/> Credit card <input type="checkbox"/> Money Order <input type="checkbox"/> Other                        |                                                                                                                                                                                     |

| FEE CALCULATION (continued)                                      |                       |                       |                       |                                                                            |          |                      |
|------------------------------------------------------------------|-----------------------|-----------------------|-----------------------|----------------------------------------------------------------------------|----------|----------------------|
| 3. ADDITIONAL FEES                                               |                       |                       |                       |                                                                            |          |                      |
| Fee Code                                                         | Large Entity Fee (\$) | Small Entity Fee Code | Small Entity Fee (\$) | Fee Description                                                            | Fee Paid |                      |
| 105                                                              | 130                   | 205                   | 65                    | Surcharge - late filing fee or oath                                        |          |                      |
| 127                                                              | 50                    | 227                   | 25                    | Surcharge - late provisional filing fee or cover sheet                     |          |                      |
| 139                                                              | 130                   | 139                   | 130                   | Non-English specification                                                  |          |                      |
| 147                                                              | 2,520                 | 147                   | 2,520                 | For filing a request for ex parte reexamination                            |          |                      |
| 112                                                              | 920*                  | 112                   | 920*                  | Requesting publication of SIR prior to Examiner action                     |          |                      |
| 113                                                              | 1,840*                | 113                   | 1,840*                | Requesting publication of SIR after Examiner action                        |          |                      |
| 115                                                              | 110                   | 215                   | 55                    | Extension for reply within first month                                     |          |                      |
| 116                                                              | 390                   | 216                   | 195                   | Extension for reply within second month                                    |          |                      |
| 117                                                              | 890                   | 217                   | 445                   | Extension for reply within third month                                     |          |                      |
| 118                                                              | 1,390                 | 218                   | 695                   | Extension for reply within fourth month                                    |          |                      |
| 128                                                              | 1,890                 | 228                   | 945                   | Extension for reply within fifth month                                     |          |                      |
| 119                                                              | 310                   | 219                   | 155                   | Notice of Appeal                                                           |          |                      |
| 120                                                              | 310                   | 220                   | 155                   | Filing a brief in support of an appeal                                     |          |                      |
| 121                                                              | 270                   | 221                   | 135                   | Request for oral hearing                                                   |          |                      |
| 138                                                              | 1,510                 | 138                   | 1,510                 | Petition to institute a public use proceeding                              |          |                      |
| 140                                                              | 110                   | 240                   | 55                    | Petition to revive - unavoidable                                           |          |                      |
| 141                                                              | 1,240                 | 241                   | 620                   | Petition to revive - unintentional                                         |          |                      |
| 142                                                              | 1,240                 | 242                   | 620                   | Utility issue fee (or reissue)                                             |          |                      |
| 143                                                              | 440                   | 243                   | 220                   | Design issue fee                                                           |          |                      |
| 144                                                              | 600                   | 244                   | 300                   | Plant issue fee                                                            |          |                      |
| 122                                                              | 130                   | 122                   | 130                   | Petitions to the Commissioner                                              |          |                      |
| 123                                                              | 50                    | 123                   | 50                    | Processing fee under 37 CFR 1.17(q)                                        |          |                      |
| 126                                                              | 180                   | 126                   | 180                   | Submission of Information Disclosure Stmt                                  |          |                      |
| 581                                                              | 40                    | 581                   | 40                    | Recording each patent assignment per property (times number of properties) | 40       |                      |
| 146                                                              | 710                   | 246                   | 355                   | Filing a submission after final rejection (37 CFR § 1.129(a))              |          |                      |
| 149                                                              | 710                   | 249                   | 355                   | For each additional invention to be examined (37 CFR § 1.129(b))           |          |                      |
| 179                                                              | 710                   | 279                   | 355                   | Request for Continued Examination (RCE)                                    |          |                      |
| 169                                                              | 900                   | 169                   | 900                   | Request for expedited examination of a design application                  |          |                      |
| Other fee (specify) _____                                        |                       |                       |                       |                                                                            |          |                      |
| **or number previously paid, if greater; For Reissues, see above |                       |                       |                       |                                                                            |          | SUBTOTAL (3) (\$) 40 |

| FEE CALCULATION       |                       |                       |                       |                                                            |                  |           |
|-----------------------|-----------------------|-----------------------|-----------------------|------------------------------------------------------------|------------------|-----------|
| 1. BASIC FILING FEE   |                       |                       |                       |                                                            |                  |           |
| Large Entity Fee Code | Large Entity Fee (\$) | Small Entity Fee Code | Small Entity Fee (\$) | Fee Description                                            | Fee Paid         |           |
| 101                   | 710                   | 201                   | 355                   | Utility filing fee                                         | 710              |           |
| 106                   | 320                   | 206                   | 160                   | Design filing fee                                          |                  |           |
| 107                   | 490                   | 207                   | 245                   | Plant filing fee                                           |                  |           |
| 108                   | 710                   | 208                   | 355                   | Reissue filing fee                                         |                  |           |
| 114                   | 150                   | 214                   | 75                    | Provisional filing fee                                     |                  |           |
| SUBTOTAL (1) (\$) 710 |                       |                       |                       |                                                            |                  |           |
| 2. EXTRA CLAIM FEES   |                       |                       |                       |                                                            |                  |           |
| Total Claims          | <u>12</u>             | -20** =               | <u>  </u>             | X                                                          | Fee from below = | <u>0</u>  |
| Independent Claims    | <u>1</u>              | -3** =                | <u>  </u>             | X                                                          | Fee from below = | <u>0</u>  |
| Multiple Dependent    |                       |                       |                       |                                                            | Fee from below = | <u>  </u> |
| Large Entity Fee Code | Large Entity Fee (\$) | Small Entity Fee Code | Small Entity Fee (\$) | Fee Description                                            | Fee Paid         |           |
| 103                   | 18                    | 203                   | 9                     | Claims in excess of 20                                     |                  |           |
| 102                   | 80                    | 202                   | 40                    | Independent claims in excess of 3                          |                  |           |
| 104                   | 270                   | 204                   | 135                   | Multiple dependent claim, if not paid                      |                  |           |
| 109                   | 80                    | 209                   | 40                    | ** Reissue independent claims over original patent         |                  |           |
| 110                   | 18                    | 210                   | 9                     | ** Reissue claims in excess of 20 and over original patent |                  |           |
| SUBTOTAL (2) (\$) 0   |                       |                       |                       |                                                            |                  |           |

| SUBMITTED BY      |                        |                                   |        | Complete (if applicable) |                |
|-------------------|------------------------|-----------------------------------|--------|--------------------------|----------------|
| Name (Print/Type) | Thomas A. Jolly        | Registration No. (Attorney/Agent) | 39,241 | Telephone                | (509) 324-9256 |
| Signature         | <i>Thomas A. Jolly</i> |                                   |        | Date                     | 11-22-2008     |

**WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Improved DVD Navigator And Application  
Programming Interfaces (APIs)**

Inventors:

**Glenn F. Evans**  
**Alok Chakrabarti**

ATTORNEY'S DOCKET NO. MS1-688US

2004/01/20 10:00:00

00224460

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

**RELATED APPLICATIONS**

This patent application is related to co-pending U.S. Patents Numbers \_\_\_\_\_, (Attorney’s Docket Numbers: MS1-703US, MS1-706US, MS1-707US, MS1-708US, and MS1-709US).

**TECHNICAL FIELD**

This invention relates to computers and like devices, and more particularly to improved methods and arrangements associated with a generic navigator program and the application programming interfaces (APIs) exposed thereby.

**BACKGROUND**

A digital versatile disc (DVD) player is composed of three logical units, as defined in the DVD specification. The first logical unit is a DVD player application that presents an interface to the user and relays user commands to the second logical unit. The second logical unit a DVD navigator that reads and interprets the information on the DVD and controls which segments of video and audio are processed based on the user commands. The third logical unit is a DVD presentation layer that decompresses data read from the DVD and presents the corresponding audio, video and subpicture streams, as applicable, to one or more renderers.

These logical units may be implemented in hardware and/or software. By way of example, in certain implementations, the DVD player is implanted via a graphical user interface (GUI) that is displayed to a user, and through which the user is able to selectively control playback, etc., of the DVD using a pointing

1 selection input device, e.g., a mouse. This is usually a fairly straightforward task  
2 for system developers and allows for easy customization.

3 Implementing a DVD navigator, on the other hand, tends to be a more  
4 complex task. This is especially true for applications that seek to integrate DVD  
5 information into presentations and the like. Here, each developer entity would  
6 need to provide a mechanism for reading and interpreting their DVD, and  
7 interfacing with the decoder mechanism in the DVD presentation layer. Moreover,  
8 the decoder mechanism in the DVD presentation layer will likely be a product of a  
9 third party; making the task of authoring a DVD navigator even more difficult,  
10 since the navigator must interface to many may have different decoder  
11 mechanisms.

12 Consequently, there is a need for a powerful yet simplified and consistent  
13 interface that player applications can use to control the DVD navigator program.

## 14 SUMMARY OF THE INVENTION

15  
16 Recognizing the potential burdens placed on application developers,  
17 Microsoft Corporation, in an effort to further enhance their operating system and  
18 the user's environment have developed a generic navigator component. This  
19 generic navigator component provides a standard, specification-compliant DVD  
20 navigator as part of Windows® to help application developers avoid such possibly  
21 repetitive and difficult tasks. This generic navigator component exposes two  
22 application programming interfaces (APIs) that combined provide a powerful, yet  
23 simplified and consistent interface that player applications can use to control the  
24  
25

1 DVD navigator. The APIs have been designed to further influence the flexibility  
2 and usefulness of the underlying DVD Navigator.

3 In accordance with certain aspects of the present invention, enhancements  
4 have been developed to further extend the performance of the generic navigator  
5 component. Of significance herein, was the need for improved APIs for use with a  
6 generic navigator program. Here, the navigator program has been enhanced to  
7 better synchronize with the player application, thereby providing better control  
8 mechanisms for the player application, improved enforcement of  
9 restricted/parental controlled content viewing, generation and assignment of  
10 unique identifiers to the content source, and an improved user and player  
11 application environment for starting and stopping playback.

12  
13  
14  
**BRIEF DESCRIPTION OF THE DRAWINGS**

15  
16 A more complete understanding of the various methods and arrangements  
17 of the present invention may be had by reference to the following detailed  
18 description when taken in conjunction with the accompanying drawings wherein:

19 Fig. 1 is a block diagram depicting an exemplary DVD player device.

20 Fig. 2 is a block diagram of a computer environment suitable for use with  
21 the DVD player device in Fig. 1.

22 Fig. 3 is. a block diagram depicting a first mode of synchronization  
23 between a DVD player application and a generic navigator program.

24 Fig. 4 is. a block diagram depicting a second mode of synchronization  
25 between a DVD player application and a generic navigator program.

1 Fig. 5 is. a block diagram depicting a third mode of synchronization  
2 between a DVD player application and a generic navigator program.

3 Fig. 6 is. a block diagram depicting a fourth mode of synchronization  
4 between a DVD player application and a generic navigator program.

5 Figs 7 and 8 are block diagrams depicting non-blocking and blocking  
6 modes of synchronization, respectively, between a DVD player application and a  
7 generic navigator program.

8 Fig. 9 is a block diagram depicting exemplary read/write communication  
9 functionality between a player application and a program related to media content.

10 Fig. 10 is a line diagram depicting a dual-branch playback decision point  
11 associated with restricted/parental control over media content.

12 Fig. 11 is a line diagram depicting a multiple-branch playback decision  
13 point associated with restricted/parental control over media content.

14 Fig. 12 is a block diagram depicting an exemplary method for controlling  
15 access to media data through the use of a player application supplied code.

16 Fig. 13 is a block diagram depicting exemplary media content bookmarking  
17 functionality.

18 Fig. 14 is an illustrative diagram depicting an exemplary method for  
19 generating a substantially unique identifier for a media source.

20  
21 **DETAILED DESCRIPTION**

22 The following exemplary methods and arrangements describe certain  
23 enhancements and features associated with a generic DVD navigator having APIs  
24 exposed to DVD player applications. These are referred to as the DVD navigator  
25

1 and DVD2 APIs. It is noted that while most of the description is directed towards  
2 a PC running the Windows® operating system, the various methods and  
3 arrangements are clearly applicable to other operating systems, devices, etc.  
4 Moreover, the use of the term DVD is not meant to exclude other media formats.  
5 Thus, the DVD content itself may come from a hard drive, a compact disc, over a  
6 network, and the like.

7 As will be described, the DVD navigator and/or DVD2 API enable a player  
8 application to interactively control the playback of DVD content. The DVD2 API  
9 consists of two interfaces. The first is termed "IDvdInfo2". The second is termed  
10 "IDvdControl2". The player application may use the IDvdInfo2 interface to query  
11 the current state of the DVD navigator and the IDvdControl2 interface to better  
12 control playback and/or to alter the DVD navigator's state.

13 The DVD2 API provides several unique and novel features. For example,  
14 thread-based synchronization methods are provided for real-time playback; a  
15 playback control mechanism is provided to determine the degree of interactivity;  
16 communication mechanisms are provided between the player application and the  
17 disc program, playing of time ranges is supported; mechanisms are provided for  
18 coordinating and handling parental level requests and for determining the minimal  
19 parental level to play a restricted segment of content; and, a unique disc identifier  
20 algorithm is provided, which further supports the bookmarking of any location  
21 within the DVD content

22 With this mind, attention is drawn to Fig. 1, which depicts an exemplary  
23 DVD player 100. Player 100 includes at least one player application 102  
24 configured to present the user with a user interface (U/I) 104. Through U/I 104,  
25



1 the user is able to instruct player application 102 with regard to the playback of  
 2 DVD content 110.

3 As illustrated, player application 102 is provided with DVD2 API 108a and  
 4 108b to communicate user requests, and receive feedback information,  
 5 respectively. DVD2 API 108a-b provide access to the functions within navigator  
 6 106. Navigator 106 interacts with DVD content 110, which in addition to media  
 7 information includes a program 112. Program 112 defines the menus, jumps, etc.,  
 8 associated with the remaining content. Navigator 106 includes a state 114  
 9 associated with the playback process. Here, in state 114, for example, the current  
 10 user operation (UOP) (e.g., play, stop, pause, reverse, fast-forward, slow motion,  
 11 angle, etc.) is stored along with the current location within the DVD content (e.g.,  
 12 chapter, time, frame) and certain other registers such as those that could record  
 13 recent jumps/UOPs.

14 The output of navigator 106 includes an encoded video stream, an encoded  
 15 audio stream, and a subpicture stream, as applicable. These outputs are inputted to  
 16 a decoder 116, which is configured to decode (decrypt and decompress) the  
 17 encoded data and output the corresponding streams to the applicable video  
 18 renderer 118 or audio renderer 120. Renderer 118 causes the video information to  
 19 be displayed to the user, for example, via a video monitor. Renderer 120 causes  
 20 the audio information to be reproduced for the listener, for example, via one or  
 21 more speakers.

22 Attention is now drawn to Fig. 2, which is a block diagram depicting an  
 23 exemplary computing system 200 suitable for use with the arrangement in Fig. 1.

24 Computing system 200 is, in this example, in the form of a personal  
 25 computer (PC), however, in other examples computing system may take the form

1 of a dedicated server(s), a special-purpose device, an appliance, a handheld  
 2 computing device, a mobile telephone device, a pager device, etc.

3 As shown, computing system 200 includes a processing unit 221, a system  
 4 memory 222, and a system bus 223. System bus 223 links together various system  
 5 components including system memory 222 and the processing unit 221. System  
 6 bus 223 may be any of several types of bus structures including a memory bus or  
 7 memory controller, a peripheral bus, and a local bus using any of a variety of bus  
 8 architectures. System memory 222 typically includes read only memory (ROM)  
 9 224 and random access memory (RAM) 225. A basic input/output system 226  
 10 (BIOS), containing the basic routine that helps to transfer information between  
 11 elements within computing system 200, such as during start-up, is stored in ROM  
 12 224. Computing system 200 further includes a hard disk drive 227 for reading  
 13 from and writing to a hard disk, not shown, a magnetic disk drive 228 for reading  
 14 from or writing to a removable magnetic disk 229, and an optical disk drive 30 for  
 15 reading from or writing to a removable optical disk 231 such as a CD ROM or  
 16 other optical media. Hard disk drive 227, magnetic disk drive 228, and optical  
 17 disk drive 230 are connected to system bus 223 by a hard disk drive interface 232,  
 18 a magnetic disk drive interface 233, and an optical drive interface 234,  
 19 respectively. These drives and their associated computer-readable media provide  
 20 nonvolatile storage of computer readable instructions, data structures, computer  
 21 programs and other data for computing system 200.

22 A number of computer programs may be stored on the hard disk, magnetic  
 23 disk 229, optical disk 231, ROM 224 or RAM 225, including an operating system  
 24 235, one or more application programs 236, other programs 237, and program data  
 25 238.

1 A user may enter commands and information into computing system 200  
2 through various input devices such as a keyboard 240 and pointing device 242  
3 (such as a mouse). A camera/microphone 255 or other like media device capable  
4 of capturing or otherwise outputting real-time data 256 can also be included as an  
5 input device to computing system 200. The real-time data 256 can be input into  
6 computing system 200 via an appropriate interface 257. Interface 257 can be  
7 connected to the system bus 223, thereby allowing real-time data 256 to be stored  
8 in RAM 225, or one of the other data storage devices, or otherwise processed.

9 As shown, a monitor 247 or other type of display device is also connected  
10 to the system bus 223 via an interface, such as a video adapter 248. In addition to  
11 the monitor, computing system 200 may also include other peripheral output  
12 devices (not shown), such as speakers, printers, etc.

13 Computing system 200 may operate in a networked environment using  
14 logical connections to one or more remote computers, such as a remote computer  
15 249. Remote computer 249 may be another personal computer, a server, a router,  
16 a network PC, a peer device or other common network node, and typically  
17 includes many or all of the elements described above relative to computing system  
18 200, although only a memory storage device 250 has been illustrated in Fig. 2.

19 The logical connections depicted in Fig. 2 include a local area network  
20 (LAN) 251 and a wide area network (WAN) 252. Such networking environments  
21 are commonplace in offices, enterprise-wide computer networks, Intranets and the  
22 Internet.

23 When used in a LAN networking environment, computing system 200 is  
24 connected to the local network 251 through a network interface or adapter 253.  
25 When used in a WAN networking environment, computing system 200 typically

1 includes a modem 254 or other means for establishing communications over the  
2 wide area network 252, such as the Internet. Modem 254, which may be internal  
3 or external, is connected to system bus 223 via the serial port interface 246.

4 In a networked environment, computer programs depicted relative to the  
5 computing system 200, or portions thereof, may be stored in the remote memory  
6 storage device. It will be appreciated that the network connections shown are  
7 exemplary and other means of establishing a communications link between the  
8 computers may be used.

9 DVD2 API 108a-b simplifies application authoring, adds functionality and  
10 solves many difficult synchronization issues common to DVD player applications  
11 development. Basically, a common DVD API helps discourage proprietary single-  
12 use monolithic DVD solutions that serve only as standalone DVD player  
13 applications. It also allows various applications (such as presentation programs,  
14 DVD players, games, or interactive learning programs) to add DVD support  
15 without having to know which DVD decoder or DVD hardware support is on the  
16 user's system. Historically, custom DVD solutions tend to be very hardware  
17 dependent and have limited upgrade options for users.

18 As will be described in greater detail below, DVD2 API 108a-b adds  
19 flexible synchronization mechanisms for the application to know the completion  
20 status of requests made to the DVD Navigator 106. The new command completion  
21 notification allows the application to concurrently perform other tasks and be  
22 informed of the status of a previous request. Previous DVD APIs assumed that  
23 either the application would be blocked until the request was completed, or would  
24 not send any notification to the application. Applications now have the option of  
25

1 receiving a synchronization object that they can use to wait on or are notified  
 2 about completion events.

3 The synchronization mechanism also returns the status of the request that  
 4 indicates whether it succeeded or returns the reason (an error code) for its failure.  
 5 Previous DVD APIs would appear to successfully execute requests that would  
 6 later fail due to changed state when the DVD Navigator 106 actually started  
 7 processing them. At that point, there was no way to propagate the error indication  
 8 back to the player application 102. The new mechanism also notifies the player  
 9 application 102 of every request that is cancelled or overridden by the disc's  
 10 program 112 or by further user actions.

11 Current DVD APIs use predefined behaviors that dictate how a command  
 12 interacts with the current display. When a player application issues a new request,  
 13 it pre-empts and cancels any content (video or audio) that is being played.  
 14 Alternatively, the APIs semantics dictate that the current presentation completes  
 15 before the new content is presented which forces the user to wait before he/she can  
 16 request another action. Interactive applications such as DVD players and games  
 17 may require the first behavior (instant effect), but other applications such as a  
 18 slideshow may require the second behavior (complete the current presentation).  
 19 Since these two options are mutually exclusive, predefined API's semantics cannot  
 20 accommodate both. DVD2 API 108a-b allows player application 102 to indicate  
 21 the desired behavior via flags, and also how it interacts with the synchronization  
 22 mechanism.

23 DVD navigator 106 is configured to simulate a virtual CPU that uses an  
 24 execution state 114 (in the form of a set of memory registers 124 (see, Fig. 9)).  
 25 Previous DVD APIs allowed applications to read the contents of the registers.

1 DVD2 API 108a-b also allows player application 102 to also change the contents  
2 of the memory registers. The combined read/write functionality allows player  
3 application 102 to essentially 'communicate' with program 112, as illustrated in  
4 Fig. 9.

5 The read and write methods works in such a way that they can also be used  
6 for synchronization. By way of example, with read/write functionality, player  
7 application 102 can implement 'controlled unlocking' or restricted access to all or  
8 portions of DVD content 110. With controlled unlocking, the user may be  
9 restricted from viewing portions of the disc until player application 102 sets  
10 specific memory registers. Player application 102 could receive this information  
11 from the content's author, the user, another program, a website, or the like. For  
12 example, Fig. 12 depicts the use of a code being written to registers 124 by player  
13 application 102 and being read by program 112. If the code is correct, then  
14 portion 130 of DVD content 110 can be played back.

15 In certain implementations, DVD2 API 108a-b contains a simplified  
16 naming scheme for the potential user operations suggested in the DVD  
17 specification Annex J. The DVD2 API uses less DVD jargon and features a more  
18 intuitive naming scheme. The user operation names proposed in the DVD  
19 specification are unclear and can lead to incorrect usage or under-utilization by  
20 application programs. The names now suggest their usage instead of an abstract  
21 label. Also time codes are now returned in a simple integer format instead of the  
22 awkward BCD coding.

23 Some previous DVD APIs failed to correctly handle minimum parental  
24 level branching by having the DVD navigator send an error event indicating that  
25 the branch always failed (see Fig. 10). The player application then had to increase

1 the parental level and restart the movie from the beginning. If the branch fails, the  
2 player application would need to stop the playback to enter the STOP domain to  
3 change the parental level. It can only continue by restarting the movie.

4 To the contrary, DVD2 API 108a-b has a mode that pauses navigator 106  
5 and lets player application 102 respond to the parental level increase request  
6 before the navigator 106 continues. If the increase request is granted, the playback  
7 continues without requiring the user to start the movie from the beginning. The  
8 DVD specification only states that the navigator should pause until it knows  
9 whether the request succeeded or failed. It does not describe a mechanism to  
10 accomplish this task and suggests that the Navigator "calls the Temporary Parental  
11 Level Change feature built into the player" (4.6.4.1 V14-197).

12 Nor does the DVD specification describe any mechanism to allow the user  
13 to play multi-segment parent level branches (see, e.g., Fig 11). As such, previous  
14 DVD APIs did not provide a mechanism that allowed the user to play multi-  
15 segment (or multiple-branch) parent level branches if no branches were permitted  
16 at the current user level. In the past, the navigator only notified the application  
17 that the playback has stopped, since no branch was available for the current  
18 parental level.

19 To the contrary, navigator 106 and DVD2 API 108a-b compute the  
20 minimum level required to play the block and return this value along with a  
21 'playback stopped' notification. The application can then notify the user of the  
22 required parental level that is required to continue playing DVD content 110.  
23 Thus, the user no longer has to guess the required level through trial and error,  
24 having to restart the movie on each try.

25

1           Additionally, DVD2 API 108a-b extends the functionality of the DVD  
 2 Annex J specification and previous DVD APIs. The DVD Annex J specification  
 3 only specifies actions to perform. It does not specify how player application 102  
 4 finds out information about the disc or the DVD navigator's state 114. Here, new  
 5 disc and navigation state query functionality is provided.

6           Unlike previous DVD APIs, DVD2 API 108a-b does not require the  
 7 application writer to already have a ready copy of the DVD specification to use it  
 8 (e.g., due to the incomplete description of the data returned by the API). The data  
 9 returned by the methods to get the textual information, the title attributes, audio  
 10 attributes and subpicture attributes is documented so that application developers  
 11 can get the necessary information from the new API and the associated  
 12 documentation.

13           DVD2 API 108a-b also allows the application to query the attributes of  
 14 arbitrary title indices instead of just the current title index. DVD2 API 108a-b also  
 15 returns the audio stream's Karaoke information so that intelligent Karaoke  
 16 applications can be implemented. DVD2 API 108a-b also returns the capabilities  
 17 of decoder 116 so the application can present configuration options to the user  
 18 (like frame stepping in both direction, smooth rewind and fast-forward etc.) or  
 19 intelligently alter the user interface. New control functionality is also provided.  
 20 For example, DVD2 API 108a-b allows player application 102 to play ranges of  
 21 chapters or ranges of times, to select specific menu buttons (just not relative  
 22 buttons) and allows the user to select buttons using a mouse location. It also  
 23 supports the getting/setting of bookmark objects and the ability to query a  
 24 calculated current unique disc ID.

25





00000000 " 00000000

1 Before describing further details of these various models and the DVD2  
2 API 108a-b, the deficiencies of a blocking-only API or a non-blocking-only API  
3 will be described. One variation is depicted in Fig. 7. Here, player application  
4 102 sends a request to navigator 106 (via DVD2 API 108a, of course). The player  
5 application 102 must wait for a result message from navigator 106. One drawback  
6 to this model is that U/I 104 will probably be "frozen" while player application  
7 102 waits.

8 One way to solve the frozen U/I problem is to provide a worker program,  
9 such as is depicted in Fig. 8. Here, the worker program receives the request and  
10 forwards it to navigator 106 and then itself waits for the result message. Once the  
11 worker receives the result message then it is forwarded to player application 102.  
12 While this may free up U/I 104, it may be difficult to manage several workers  
13 operating simultaneously.

14 In contrast, a non-blocking API is equivalent to the "don't care" mode.  
15 There is no direct feedback on the status or result of an operation. The application  
16 must infer the status from changes in the playback (time changes, menu changes,  
17 etc). However, due to variation in disc content and structure, this approach is very  
18 unreliable and error prone. With this mind, the following sections provide  
19 additional details into the use of DVD2 API 108a-b

20 All of the IDVDCControl methods in previous DVD APIs run  
21 asynchronously to the application (a non-blocking-only model). Thus, when an  
22 application 102 calls a method, the navigator 106 performs preliminary  
23 verifications and then immediately returns a result. However, in the meantime, the  
24 state of the DVD Navigator may have changed and the request may fail when the  
25 DVD Navigator actually begins to execute the command.

1           One solution is to change the semantics of the DVD API to ensure that  
 2 methods do not return until all requests complete. But to retain the asynchronous  
 3 behavior, applications must create separate execution paths (e.g., helper threads) to  
 4 manage DVD API calls (as described above in a blocking-only model).  
 5 Multithreaded programming models always complicate application development,  
 6 especially simple scriptable interfaces.

7           Therefore, to solve this problem, the DVD2 API 108a-b creates associated  
 8 synchronization command objects. The command object allows the application to  
 9 synchronize and to learn about the command's status. Each API method is  
 10 extended with two extra arguments. The general form of a DVD2 API command  
 11 is:

12 HRESULT IDVdControl2::Command( arguments, dwFlags, IDvdCmd\*\* ppObj)

13           Wherein: ppObject is an argument used to return a synchronization COM  
 14 (Component Object Model) object to application 102; and, dwFlags is the set of  
 15 flags passed to the method to determine the behavior and usage of the  
 16 synchronization object. These are a bit-wise union of the available pre-defined  
 17 flags.

18           The synchronization object has the following interface:

```
19 interface IDvdCmd : IUnknown
20 {
21     HRESULT WaitForStart();
22     HRESULT WaitForEnd();
23 }
```

24           The object returned must be released by the application. By returning a pre-  
 25 incremented COM object, the life of the object can be correctly maintained. A

1 variation on the interface also extends the original interface by including two  
 2 methods that allow the application to wait on the start and end occurrence along  
 3 with other changes in the system:

```
4 HANDLE GetStartHandle( );
5 HANDLE GetEndHandle( );
```

6  
 7 The flags take the following values:

8 DVD\_CMD\_FLAG\_SendEvents - events are sent regarding the request's  
 9 status

10 DVD\_CMD\_FLAG\_Block - do not continue until the command has been  
 11 completed

12 DVD\_CMD\_FLAG\_None - a placeholder indicating no flags

13 The special return code VFW\_E\_DVD\_CMD\_CANCELLED is returned  
 14 by the initial DVD API method, by the IDvdCmd::WaitForStart or  
 15 IDvdCmd::WaitForEnd or along with the event indication that the command was  
 16 pre-empted and is no longer valid.

17 A sample example of C++ usage of a command object is as follows:

```
18 IDvdCmd* pObj ;
19 HRESULT hres = IDvdControl2->PlayTitle (15, DVD_CMD_FLAG_None ,&pObj );
20 // don't wait or notify
21 pObj ->Release ( ) ;
```

22 As described above, player application 102 can determine the  
 23 commencement and completion of the command, by any of the following: using  
 24 the command object directly, using no command objects, listening to command  
 25

1 related events, using a combination of events and objects to aid in tracking  
 2 multiple instances of a command.

3 **Using objects**

4 By passing an IDvdCmd pointer to the command, the Navigator will  
 5 allocate and return a new IDvdCmd object. Calling the interface method  
 6 IDvdCmd::WaitForStart() will block until the command begins and  
 7 IDvdCmd::WaitForEnd() waits until the command completes. If the command  
 8 has been cancelled, then the Navigator will return  
 9 VFW\_E\_COMMAND\_CANCELLED. After the application is done with the  
 10 object, it must call Release() to free the COM object. A NULL pointer passed to  
 11 the DVD API indicates that no command object should be returned to the  
 12 application and the command execution should continue in the standard  
 13 asynchronous mode.

14 The other two methods GetStartHandle() and GetEndHandle() return a  
 15 system specific synchronization object that allows the application to wait for other  
 16 requests (disc I/O, user interface changes, semaphore changes, unblocking threads,  
 17 communications with other processes, etc) to be processed while it wait for the  
 18 start or end events to occurs. Then the application calls the WaitForStart() or  
 19 WaitForEnd() methods to retrieve the result. An example in the Microsoft  
 20 Windows API:

```

  21     handleStart = GetStartHandle()
  22     Signaled = WaitForMultipleObjects( handleDiscIO, handleUserInter, ..., handleStart )
  23     If signaled = handleStart
  24         Result = DvdCmd->WaitForStart()
  
```

25

1 **Not using Objects**

2 Instead of managing an object, the application can simply specify the  
 3 DVD\_CMD\_FLAG\_Block flag with a null object pointer. The command will not  
 4 return until it has either completed or was cancelled. The API will emulate a  
 5 synchronous behavior. For example:

```
6 HRESULT hres = IDvdControl2->PlayTitle(uTitle, DVD_CMD_FLAG_Block,0);
```

7 is semantically equivalent to:

```
8 IDvdCmd* pObj ;
9 HRESULT hres = IDvdControl2->PlayTitle( uTitle,
10 DVD_CMD_FLAG_Block, &pObj);
11 If( succeeded ( hres) ) {
12     Hres = pObj->WaitToEnd();
13     pObj->Release();
14 }
15
```

11 **Using Events**

12 Specifying the DVD\_CMD\_FLAG\_SendEvents flag will cause the  
 13 Navigator to issue the following events:

```
14 {EC_DVD_CMD_START, lParam1, HRESULT}
15 {EC_DVD_CMD_END, lParam1, HRESULT}
```

16 If an application only needs to synchronize one command (or does not  
 17 differentiate between command instances), no synchronization object is needed  
 18 and only events are required. A NULL object pointer is passed to the DVD API  
 19 method and the lParam1 value sent with the event will always be set to 0.

20 **Using Events and Objects**

21 By specifying both objects and the DVD\_CMD\_FLAG\_SendEvents flag,  
 22 an application can track different commands. The DVD2 API call will return an  
 23 object that the application can use for later reference. When the event notification  
 24 is sent, the DVD2 API generates a unique identifier (or 'cookie') lParam1 for each  
 25

1 event that the application can map back to an IDvdCmd object. The cookie  
 2 approach ensures that applications will not leak memory if they miss an event and  
 3 allows the DVD Navigator to verify the validity of the object.

4 The DVD2 API method IDvdInfo2::GetCmdFromEvent( IParam1 ) maps  
 5 the cookie into a command object pointer. The application must call the COM  
 6 "Release" method on the returned pointer after it has finished processing each of  
 7 these events. When the application is completely finished with the message  
 8 (usually after receiving an END event), it must call "Release" on the global  
 9 command pointer that it saved.

10 **Example of Blocking/Non-Blocking**

11 The following illustrative examples show how synchronization can be  
 12 accomplished using the IDvdControl2 interface:

13 For clarity, some of the examples refer to the following utility function used  
 14 to map the IParam1 value from EC\_DVD\_CMD events into an IDvdCmd object:

```

15 IDvdCmd* GetDvdCmd( LONG_PTR IParam )
16 {
17     IDvdCmd* pCmd;
18     plDvdInfo2->GetCmdFromEvent (iParam, &pCmd) ;
19     return pCmd;
20 }
  
```

18 **No synchronization (Asynchronous model)**

19 The application calls the method to request an action:

```

20 HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, NULL);
  
```

21 **Synchronization without events**

22 An example of the correct way to wait for a command to end without using  
 23 events is:

```

24 IDvdCmd* pObj ;
25 HRESULT hres = IDvdControl2->PlayTitle( uTitle, 0, &pObj);
  
```





```

1 DVD_CMD_FLAG_SendEvents, &pGlobalObj );
2
3 // (*1)
4
5 If( FAILED ( hres)) {
6     pGlobalObj = NULL;
7 }
8 ...
9
10 In the event processing function:
11
12 Function ProcessEvent( type, IParam1, IParam2 )
13
14 switch (type)
15 {
16
17 case EC_DVD_CMD_END:
18     IDvdCmd* pObj = GetDvdCmd( IParam1 );
19
20     HRESULT hres = IParam2;
21
22     If( NULL != pObj ) {
23         // if the object returned by the event matches the global pointer returned
24         // by the PlayTitle, process it
25         If (pGlobalObj == obj ) {
26             ProcessCmdEnd....
27             pGlobalObj ->Release ( );
28             pGlobalObj = NULL;
29         }
30         pObj ->Release ( );
31     }
32 }
33
34 break ;
35

```

## 1 Full synchronization using events and a separate event loop thread

2 An example of the correct way to wait for a command using events is:

```
3 // in global code
4 IDvdCmd* pGlobalObj=0;
5 {
6     LockCriticalSection
7     HRESULT hres = IDvdControl2->PlayTitle( uTitle,
8         DVD_CMD_FLAG_SendEvents, &pGlobalObj );
9     if( FAILED ( hres) ) {
10         pGlobalObj = NULL;
11     }
12     UnlockCriticalSection
13 }
```

9 Function ProcessEvent( type, lParam1, lParam2 )

```
10 switch (type)
11 {
12     case EC_DVD_CMD_COMPLETE:
13     case EC_DVD_CMD_CANCEL:
14     {
15         CautoLock(globalCritSect );
16         IDvdCmd* pObj = GetDvdCmd( lParam1 );
17         HRESULT hres = lParam2
18         if( NULL = pObj ) {
19             if (pGlobalObj == pObj) {
20                 pGlobalObj ->Release ( );
21                 pGlobalObj = NULL;
22             }
23             pObj ->Release ( );
24         }
25         break ;
26     }
27 }
```

## 18 Exemplary Playback Interactivity Control Mechanism

19 Previous DVD API commands assumed that on any change of content,  
20 player application 102 wanted to truncate the current content presentation, and it  
21 switched to the new content. The improved DVD2 API commands extend the  
22 command object mechanism with the following flags:

23 DVD\_CMD\_FLAG\_Flush

24 DVD\_CMD\_FLAG\_StartWhenRendered

25

00000000 " 2004/02/26/00

DVD\_CMD\_FLAG\_EndAfterRendered

Here, the ..\_Flush flag indicates that the presentation of the current content should be immediately truncated so that new content can start to be displayed (like before). The absence of the flag indicates that the current content presentation should end first. The ...\_..Rendered flags change the semantics of the start and end of each command. By default, the command starts and ends once it has been processed. The new flags indicate that the start and end occur when the results of the change of content have been processed and presented respectively.

**Exemplary Disc Communication Mechanism**

DVD2 API 108a-b permits player applications not only to read the DVD Navigator's general purpose registers (the GPRMs), but also allows them to set the GPRMs using:

```
IDvdInfo2::GetAllGPRMs(WORD pwRegisterArray[16] )
IDvdControl2::SetGPRM( ULONG ulindex, WORD wValue, DWORD dwFlags, IDvdCmd**
ppCmd )
```

The combined read/write functionality allows DVD applications to 'communicate' with the program on the disc and can implement 'controlled unlocking' or restricted access to the content. The application can use GetAllGPRMs to read the current state and set a specific register using SetGPRM.

The SetGPRM method can also be used to synchronize the application and the DVD Navigator's virtual CPU. The SetGPRM method is executed only during the periods when the DVD Navigator is allowed to process user commands (the Presentation and Still phases, 3.3.6.1 V13-28). Navigation command execution is considered to be atomic. So setting the GPRM is postponed until these phases occur. The application can use the command object and event mechanism to

1 ensure coordination. The command object's event mechanism is serialized with  
 2 event notifications (such as domain changes or changes to system registers). The  
 3 application can call SetGPRM and wait until the command completion event is  
 4 received, and then wait for an event indicating a change the DVD navigator's state  
 5 (possibly a domain change).

6 One such way to accomplish disc to application communication is  
 7 illustrated by the following pseudocode:

8 Disc sends data and awaits reply:  
 9       Disc alters a GPRM value (using a on-disc navigation command)  
 10       Disc changes its state (e.g. changes its domain)  
 11       Loops waiting for a GPRM change (caused by the application)  
 12 Application receives GPRM data and replies:  
 13       Waits for the state change (e.g. the disc's domain change)  
 14       Reads GPRM value  
 15       Sets a GPRM value using SetGPRM

16 One such way to accomplish application to disc communication is  
 17 illustrated by the following pseudocode:

18 Application sends data and awaits acknowledgement:  
 19       Application sets the data using SetGPRM  
 20       Application waits for a domain change before continuing  
 21 Disc receives data and returns acknowledgement:  
 22       Disc reads GPRM  
 23       Disc changes its state (e.g. changes its domain)

24  
 25 **Exemplary Query (Info) Interfaces**

1 Even though the DVD specification does not suggest any data retrieval  
2 methods, the DVD2 APIs do provide this capability. The following is a list of  
3 methods provided:

4 GetAllGPRMs  
5 GetAllSPRMs  
6 GetAudioLanguage  
7 GetCurrentAngle  
8 GetCurrentAudio  
9 GetCurrentButton  
10 GetCurrentDomain  
11 GetCurrentLocation  
12 GetCurrentSubpicture  
13 GetNumberOfChapters  
14 GetPlayerParentalLevel  
15 GetSubpictureLanguage  
16 GetTotalTitleTime  
17 GetTitleParentalLevels  
18 GetCurrentUOPS  
19 GetCurrentVolumeInfo (IDVD1::GetDVDVolumeInfo)  
20 GetDVDDirectory (IDVD1::GetRoot)  
21 GetAudioAttributes( [in] ULONG ulStream, [out] DVD\_AudioAttributes \*pATR );  
22 GetCurrentVideoAttributes( [out] DVD\_VideoAttributes \* pATR );  
23 GetVMGAttributes( [out] DVD\_MenuAttributes \* pATR );  
24 GetTitleAttributes( ULONG ulTitle, [out] DVD\_MenuAttributes \* pMenu, [out]  
25 DVD\_TitleAttributes\* pTitle );  
26 GetSubpictureAttributes( [in] ULONG ulStream, [out] DVD\_SubpictureAttributes  
27 \*pATR );  
28 GetButtonAtPosition( POINT point, [out] ULONG \*puButtonIndex );  
29 GetButtonRect( ULONG ulButton, RECT \*pRect );  
30 GetDefaultAudioLanguage( LCID\* pLanguage, DVD\_AUDIO\_LANG\_EXT\*  
31 pAudioExt );  
32 GetDefaultMenuLanguage( LCID\* pLanguage );  
33 GetDefaultSubpictureLanguage( LCID\* pLanguage,  
34 DVD\_SUBPICTURE\_LANG\_EXT\*pSubpictureExtension );  
35 GetDVDTextLanguageInfo( ULONG ulLangIndex, ULONG\* pulNumOfStrings,  
36 LCID\*pwLangCode, DVD\_TextCharSet \* pbCharacterSet );  
37 GetDVDTextNumberOfLanguages( ULONG \* pulNumOfLangs );  
38 GetDVDTextStringAsNative( ULONG ulLangIndex, ULONG ulStringIndex, BYTE\*  
39 pbBuffer, ULONG ulMaxBufferSize, ULONG\* pulActualSize, enum  
40 DVD\_TextStringType\* pType );  
41 GetDVDTextStringAsUnicode( ULONG ulLangIndex, ULONG ulStringIndex,  
42 WCHAR\*pchBuffer, ULONG ulMaxBufferSize, ULONG\* pActualSize,  
43 DVD\_TextStringType\* pType );  
44 GetCmdFromEvent( LONG\_PTR dwID, IDvdCmd\*\* ppCmd );  
45 GetDecoderCaps( DVD\_DECODER\_CAPS \*pCaps );  
46 GetDiscID( LPCWSTR pszPath, ULONGLONG\* pullUniqueID );  
47 GetKaraokeAttributes( [in] ULONG ulStream, DVD\_KaraokeAttributes \*pATR );  
48 GetMenuLanguages( LCID \*pLang, ULONG uMaxLang, ULONG \*puActualLang );

00000000 " 20041226

1 IsAudioStreamEnabled( ULONG ulStreamNum, BOOL \*pbEnabled );  
2 IsSubpictureStreamEnabled( ULONG ulStreamNum, BOOL \*pbEnabled );

## 3 Exemplary Control Interfaces

### 4 1) Period Playback Interface

5 In addition to playing ranges of chapters, the DVD2 API allows the playing  
6 of time periods using:

7 PlayPeriodInTitleAutoStop(ULONG ulTitle, DVD\_HMSF\_TIMECODE\* pStartTime,  
8 DVD\_HMSF\_TIMECODE\* pEndTime, DWORD dwFlags, IDvdCmd\*\* ppCmd )

9 With this method, applications (such as video editing programs and games)  
10 can accurately playback arbitrary portions of the content. Combined with the  
11 command object mechanism, any application like slideshow presentation, video  
12 games interludes, or kiosks can be implemented using a single DVD2 API  
13 command.

### 14 2) Default language Interfaces

15 SelectDefaultAudioLanguage(LCIDLanguage,DVD\_AUDIO\_LANG\_EXT  
16 audioExtension)

17 SelectDefaultSubpictureLanguage(LCIDLanguage, DVD\_SUBPICTURE\_LANG\_EXT  
18 subpictureExtension )

19 These methods allow applications (from user) to set the default language  
20 choices for DVD playback.

### 21 3) Button index selection

22 Applications can now automate menu navigation through the method

23 SelectButton( ULONG ulButton )

### 24 4) Bookmarking APIs

25 Applications can save and restore the entire DVD state (see bookmark  
patent)

1           GetState( IDvdState \*\*pStateData )

2           SetState( IDvdState\* pState, DWORD dwFlags, [out] IDvdCmd\* ppCmd )

3 5) Other

4           AcceptParentalLevelChange( BOOL bAccept ) – Please refer to the  
5 following "Minimum parental level branching" section.

6           SetGPRM( ULONG ulindex, WORD wValue, DWORD dwFlags, [out]  
7 IDvdCmd\*\* ppCmd ) -

8           SetOption( DVD\_OPTION\_FLAG flag, BOOL bEnable ) - extendible  
9 option setting mechanism

10  
11 **Mechanism for coordinating minimum parental level branching**

12           According to the DVD specification (section 4.6.4.1 pV14-197), when the  
13 DVD Navigator encounters a 'SetTmpPML' (set temporary parental management  
14 level) command, it should request permission from the application ("call the  
15 Temporary Parental level Change feature built into the player") to temporarily  
16 raise the current level. If the parental level change is allowed, the Navigator raises  
17 the parental level and branches to the restricted piece of content. Otherwise, it  
18 continues with the next command.

19           Under the semantics of the previous DVD API, when the DVD navigator  
20 executes a SetTmpPML instruction, it only sends a  
21 PARENTAL\_LEVEL\_TOO\_LOW event to the application. It immediately  
22 continues on executing the next command as if the parental level change failed.  
23 The application receives the event, stops the playback, displays a user interface to  
24 change the parental level, and then restarts the movie from the beginning.  
25 According to the DVD specification, the Navigator is allowed to alter the parental

1 level only when it is in the STOP Domain. As a result, since the navigator does  
 2 not pause at the change it must stop the playback.

3 With DVD2 API 108a-b, for example, the following sequence may occur.  
 4 The application notifies the API of the availability of the parental level change  
 5 feature by calling the method:

```
6 IDVDCControl2::SetOption( DVD_NotifyParentalLevelChange, TRUE)
```

7 When the DVD Navigator encounters a SetTmpPML instruction, it sends  
 8 a PARENTAL\_LEVEL\_TOO\_LOW event to the application. The application is  
 9 expected to display some user interface to let the user increase the parental level.  
 10 The DVD Navigator blocks until the application responds by calling  
 11 IDVDCControl2::AcceptParentalLevelChange() with TRUE or FALSE, and then  
 12 proceeds accordingly without having to stop the playback.

13  
 14 **Mechanism for aiding playback of multi-segment parental level branches**

15 The DVD specification (Section 4.1.4.1 V14-22) describes a scheme for  
 16 selecting different program chains (usually different possible segments of content)  
 17 based on the current parental level. For example, at a certain point in the video,  
 18 different versions of a scene could be available and are automatically selected by  
 19 the navigator based on the parental level (e.g. segments intended for PG, R rated or  
 20 children).

21 For each title, the PTL\_MAI table maps the current parental level into a 16-  
 22 bit mask. During playback, the DVD Navigator obtains the current parental bit  
 23 mask from the PTL\_MAI table. The parental bit mask is used when the Navigator  
 24 encounters a parental block (a collection of program chains in which each program  
 25 chain has an exclusive parental bit mask). The Navigator searches each



1 PTLID\_FLD in the VTS\_PGCI\_SRP (Section 4.2.3 V14-62) for a program chain  
 2 with a bit mask that shares common bits with the current parental bit mask.

3 If no program chain partially matches the current bit mask, previous  
 4 versions of the DVD Navigator would halt the playback and send a  
 5 DVD\_ERROR\_LowParentalLevel event to the application.

6 To help the user, certain exemplary implementations of DVD2 API 108a-b  
 7 uses the following algorithm to compute the minimum required parental level that  
 8 would let the user continue:

```

  9 Initialize PTL_MASK = 0 (the possible allowed parental levels)
 10 For each program chain index i in the VTS_PGCI_SRP
 11     If VTS_PGCI_SRP[i].BlockType = 1 (in a parental block)
 12         PTL_MASK = PTL_MASK union VTS_PGCI_SRP[i].PTL_ID_FLD
 13     If PTL_MASK = 0 then
 14         no parental level is present, so any level will work
 15     Else
 16         for each parental level index i in the PTL_MAI
 17             Let PTL_LVLI = PTL_MAI[8 - i]
 18             If PTL_LVLI[title_index] & PTL_MAI[8 - i] = 0
 19                 (note: title_index = 0 in the VMGM domain)
 20                 Return i
  
```

21 The index *i* is returned along with the DVD\_ERROR\_LowParentalLevel  
 22 event. The application 102 can use the index to suggest a possible parental level  
 23 setting to the user.

24  
 25 **Bookmarking**

1 DVD navigator 106 is configured to allow a player application 102 to  
2 encode and store the current state 114 of the DVD playback into an abstract object  
3 (referred to a bookmark 150) containing a persistent block of data. Fig. 13 depicts  
4 exemplary bookmarking functionality.

5 To further abstract and simplify the usage, DVD2 API 108a-b is configured  
6 to save, restore and query the state information contained in the bookmark. Player  
7 application 102 can query information in the bookmark 150 using the navigator  
8 106 and save it for later use. Player application 102 can later resume playback by  
9 instructing the DVD navigator 106 to restore the DVD playback state 114  
10 contained in the bookmark. Restoring bookmarks allows the player application to  
11 start playing from any arbitrary location, and any number of them for a DVD  
12 content 110. The bookmarks can be stored either in short term (memory) storage  
13 or long term storage (for example, a hard drive), and can be restored even after  
14 player application 102 and/or the PC has been shutdown and restarted. The  
15 bookmark not only contains the state of the DVD navigator (such as internal  
16 register values, playback location, playback state) but also the information about  
17 the current disc content being played and the user's settings. Player application  
18 102 can use this extra information to intelligently select the appropriate bookmark  
19 from previously saved ones that can be played for a particular disc (usually the  
20 disc being played), for example. Bookmarks can be also be shared between users  
21 and between various applications

22 The bookmarking abstract data type is comprised of two aspects; 1) the  
23 actual bookmark 150 itself, and 2) the API calls used to save, restore and query  
24 information contained in the bookmark. In accordance with certain exemplary  
25 implementations, bookmark 150 contains at least the following information: a

1 substantially unique disc identifier 145, the address of the current video object unit  
 2 (VOBU) being displayed (section 5.1.1 of the DVD specification), the loop count  
 3 and shuffle history (Section 3.3.3.2 of the DVD specification), the current DVD  
 4 resume information (outlined in section 3.3.3.2 of the DVD specification), the  
 5 current DVD general parameter (GPRM) and system parameter (SPRM) values  
 6 (sections 4.6.1.1 and 4.6.1.2), and the current domain and phase (section 3.3.3 and  
 7 3.3.6). In certain further implementations, the bookmark also includes versioning  
 8 and integrity information. The bookmark 150 can be packaged as an abstract  
 9 object or as a block of binary data for storage.

10 To provide such bookmarking techniques, DVD2 API 108 in certain  
 11 exemplary implementations supports the following methods:

- 12 1. To create a bookmark from the current location  
 13 Bookmark = GetBookmark()
- 14 2. To cause the DVD Navigator to change its location to the bookmark  
 15 SetBookmark( bookmark )
- 16 3. To find out the disc that a bookmark is intended for  
 17 DiscID = GetDiscIdentifierFromBookmark( bookmark )
- 18 4. To convert a bookmark to and from its binary representation:  
 19 BinaryData(data,size) = ConvertBookmarkToBinary( bookmark )  
 20 Bookmark = ConvertBinaryToBookmark( BinaryData )

21 Application pseudocode to implement storing the current location or to  
 22 implement power saving functionality (i.e. the ability to save the computer's state  
 23 to enter a low power state that can be restored):

```
24 Bookmark = GetBookmark()
25 BinaryData(data,size) = ConvertBookmarkToBinary( bookmark )
```

```

1          Store BinaryData(data,size)
2          Shutdown or enter power saving
3      On return from power saving, do the following to resume playback of
4      DVD:
5          Retrieve BinaryData(data,size)
6          Bookmark = ConvertBinaryToBookmark( BinaryData )
7          If GetDiscIdentifierFromBookmark( bookmark ) = current Disc Id
8          Then
9              SetBootmark( bookmark )

```

11 An example of pseudocode for an application to implement intelligent  
12 bookmarks

```

13          For each stored bookmark "bookmark"
14          If GetDiscIdentifierFromBookmark( bookmark ) = current Disc Id
15          Then
16              Add bookmark to the user selectable list

```

19 **Unique Identifier Generation**

20 The current DVD specification has a built-in unique identifier on each disc  
21 ("DVD unique identifier"). However, applications must assume that the disc  
22 authors correctly implemented the identifier; unfortunately, this not always so.

23 Many applications need a unique tag to identify a DVD disc, such as when  
24 a user swaps DVD discs, the playback system needs to decide if it has a new disc.  
25 If it has a new disc, then it must reset the playback, otherwise it can continue

1 without interrupting the user's viewing. If it does not have the ability to  
2 differentiate discs, it must always reset. A unique identifier 145 (see, Fig. 13)  
3 would provide the ability to differentiate different discs (not different exact copies,  
4 however).

5 A unique identifier 145 also lets applications verify the compatibility of  
6 stored information with a particular DVD disc. Applications cannot successfully  
7 use cached information with the wrong disc. For example, when a user attempts to  
8 recall a saved location on the disc using a bookmark, the DVD navigator 108 can  
9 ensure the data's compatibility by comparing the unique identifier stored in the  
10 bookmark with the unique identifier of the current disc. Playback only continues  
11 if the identifiers match.

12 Unique identifiers 145 allow applications to associate additional  
13 information with the disc by using the unique identifier as an index into a  
14 database. For example, even though the DVD specification supports textual  
15 information on the disc, it is rarely used. A web-based database of the disc's title  
16 and contents can be stored and retrieved by an application after it computes the  
17 identifier on the disc.

18 The current built-in unique identifier on the DVD disc is inadequate. First,  
19 the identifier is relatively large in size (32 bytes), it relies on the disc author to  
20 ensure that it is actually unique, and a central entity must assign ranges of  
21 identifiers to disc authors to ensure that the uniqueness is maintained between  
22 companies.

23 Other conventional "unique" identifier algorithms do not produce unique  
24 identifiers for a large numbers of discs. Here, the probability that two discs are  
25 assigned the same identifier grows exponentially as the total number of DVD discs

1 increases. With the expected growth trends in DVD discs, many 'unique' identifier  
 2 routines will be inadequate. Moreover, these algorithms often do not have known,  
 3 and/or provable properties. Without known properties, it is impossible to state the  
 4 effectiveness or suitability of the identifiers produced.

5 In accordance with certain exemplary implementations of the present  
 6 invention, a unique identifier 145 is generated by computing a 64-bit CRC of a  
 7 concatenated or otherwise arranged binary representation of the file header and the  
 8 file contents of various files in the DVD's VIDEO\_TS directory. This is capability  
 9 is further illustrated in Figs 13 and 14.

10 A UniqueID2 algorithm generates the identifier in four steps:

11 Step 1. The filenames of the VIDEO\_TS directory are collected and sorted  
 12 alphabetically.

13 Step 2. The file headers from each file are computed in the CRC.

14 Step 3. The data from the VMGI file ("VIDEO\_TS\VIDEO\_TS.IFO") is  
 15 computed in the CRC.

16 Step 4. The data from the first VTSI file ("VIDEO\_TS\ VTS\_xx\_O.IFO")  
 17 is computed in the CRC.

18  
 19 The 64-bit CRC is computed using an irreducible polynomial in the field  
 20  $GF(2)$ . An example polynomial is:

21 
$$P_{64} = x^{64} + x^{61} + x^{58} + x^{56} + x^{55} + x^{51} + x^{50} + x^{47} + x^{42} + x^{39} + x^{38} + x^{35} + x^{33}$$
  
 22 
$$+ x^{32} + x^{31} + x^{29} + x^{26} + x^{25} + x^{22} + x^{17} + x^{14} + x^{13} + x^9 + x^8 + x^6 + x^3 + x^0$$

23 The polynomial is generated by finding an exponent  $n$  such that  $x^n - 1$  has  
 24 an irreducible (prime) factor of degree 64.

25

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

The actual CRC value is computed, in certain examples, by concatenating all of the binary data into a single block (bits  $b_0$  to  $b_n$ ), assigning each bit  $b_i$  to the coefficient  $x^i$  in a polynomial, then computing the remainder after dividing by the polynomial  $P_{64}$ :

$$CRC_{64} = \left[ \sum_{i=0}^n b_i x^i \right] \text{mod } p_{64}$$

Here's an exemplary implementation:

Step 1

The filenames of the VIDEO\_TS directory are collected and sorted alphabetically in to a list.

Step 2

For each filename in the list, the following structure is filled out and added to the CRC (all data fields are in LSB first):

- Unsigned 64 bit integer: dateTime (the time elapsed in 100 nanosecond intervals from January 1, 1601)
- unsigned 32 bit integer: dwFileSize
- BYTE bFilename[ filename Length ]
- BYTE bFilenameTermNull=0

Step 3

If present, the first 65536 bytes of the file "VIDEO\_TS\VIDEO\_TS.IFO" are read and added to the CRC. If the IFO file is less than 65536, then the entire file is added.

Step 4





1     CLAIMS

2             What is claimed is:

3             1.     An apparatus comprising:

4             memory; and

5             logic operatively coupled to the memory and operatively configurable to  
6     access multimedia content from a medium, the logic providing a multimedia  
7     navigator program, a control application programming interface (API) and an  
8     information API, the control and information APIs being configured to respond to  
9     flags that selectively determine if at least one operation will be conducted, the  
10    operation being selected from a group of operations that includes a player-  
11    navigator synchronization operation, a selective interactive user operation, and a  
12    read/write register operation.

13  
14            2.     The apparatus as recited in Claim 1, wherein the player-navigator  
15    synchronization operation performs synchronizing steps that cause a multimedia  
16    player application to output a request command to the navigator program; and a  
17    multimedia content navigator program to subsequently return an event identifier  
18    and status result to the player application upon commencement, completion or  
19    cancellation of the requested command.

20  
21            3.     The apparatus as recited in Claim 2, wherein the request command  
22    and the event identifier are both communicated via at least one application  
23    programming interface (API) operatively associated with the navigator program.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

4. The apparatus as recited in Claim 3, wherein the API is further configured to respond to at least one flag value that is selectively set by the player application to identify that the event identifier and status result should be returned upon commencement, completion or cancellation of the requested command.

5. The apparatus as recited in Claim 2, wherein the navigator program is configured to operatively access multimedia information in response to the request command.

6. The apparatus as recited in Claim 5, wherein the multimedia information includes digital versatile disc (DVD) formatted content

7. The apparatus as recited in Claim 1, wherein the memory provides at least one register and the selective interactive user operation causes a multimedia player application to write data to at least one register that is operatively associated with a multimedia navigator program and allows at least one program defined within the multimedia content to read the at least one register.

8. The apparatus as recited in Claim 7, wherein the data includes a code, and the at least one program responds to the code by allowing at least a portion of a remaining multimedia content to be accessed.

9. The apparatus as recited in Claim 7, wherein the multimedia content includes digital versatile disc (DVD) formatted content.

1           10. The apparatus as recited in Claim 9, wherein the data includes  
2 precise playback information associated with the DVD formatted content.

3  
4           11. The apparatus as recited in Claim 10, wherein the precise playback  
5 information includes a title, a start time and an end time.

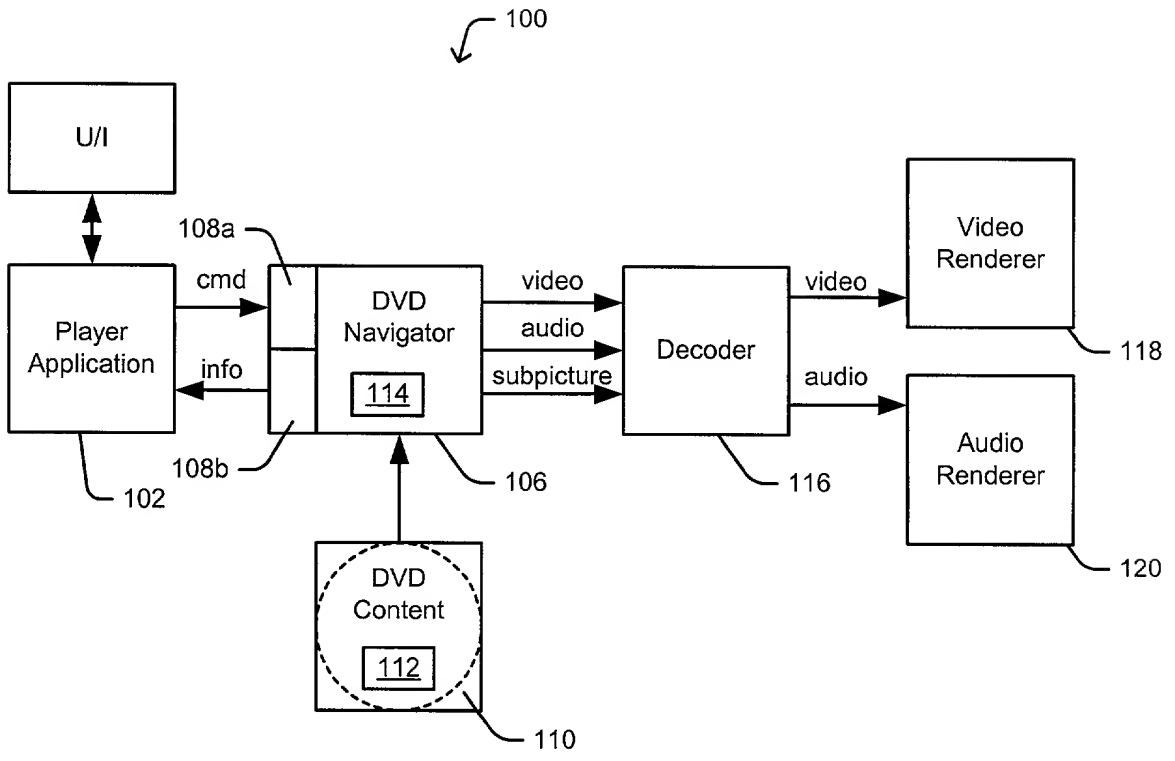
6  
7           12. The apparatus as recited in Claim 8, wherein the multimedia player  
8 application writes the data to the at least one register via at least one application  
9 programming interface (API) operatively associated with the multimedia navigator  
10 program.

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

1 **ABSTRACT**  
2

3 In accordance with certain aspects of the present invention, enhancements  
4 have been developed to further extend the performance of a generic DVD  
5 navigator component. Here, the DVD navigator program has been enhanced to  
6 better synchronize with DVD player applications. As such, better control  
7 mechanisms for the player application are provided. For example, improved  
8 enforcement of restricted/parental controlled content viewing, generation and  
9 assignment of unique identifiers to the content source, and an improved user and  
10 player application environment for starting and stopping playback are provided.  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

0000000000000000000000000000000000



*Fig. 1*

0000044260

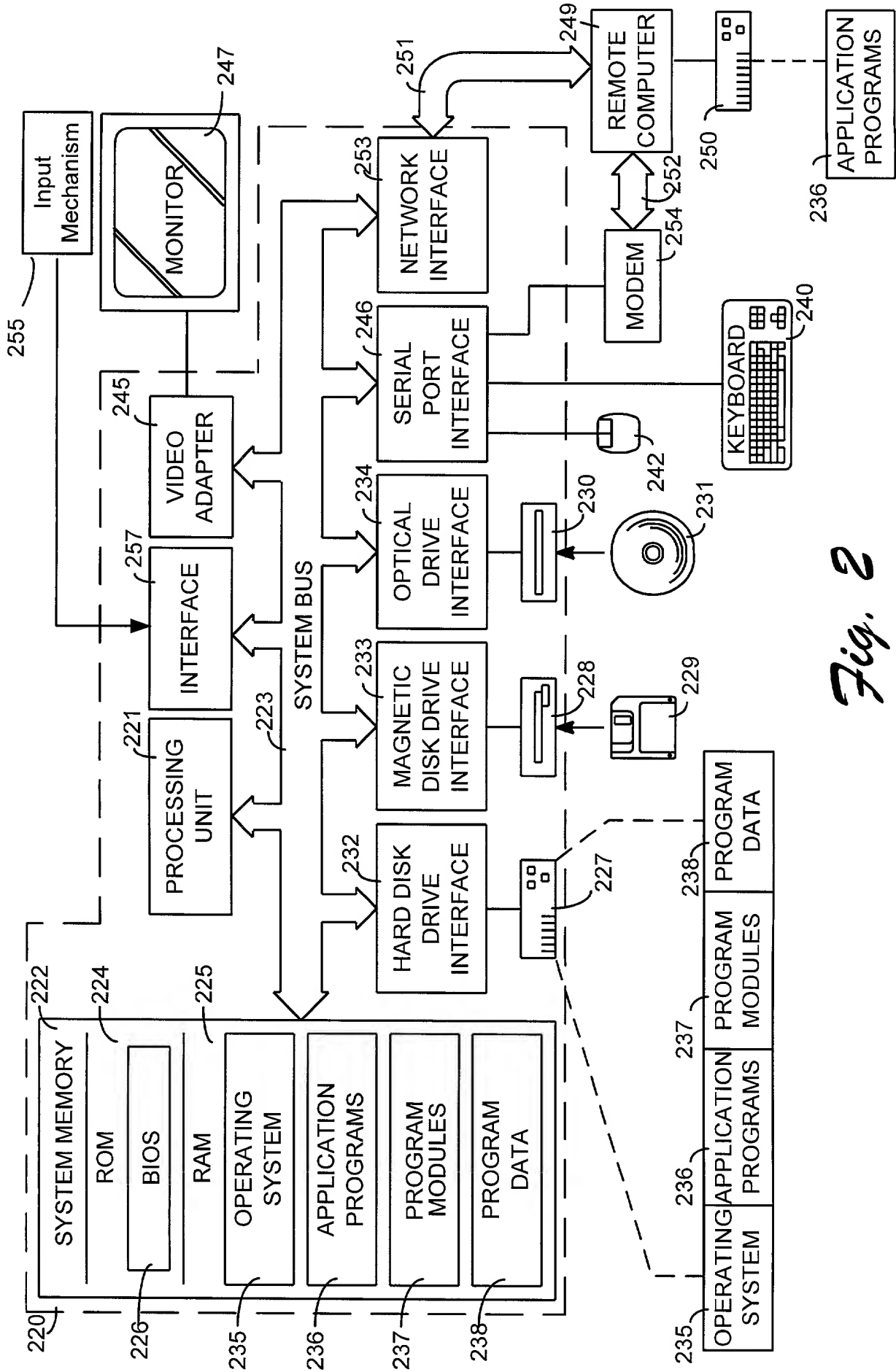
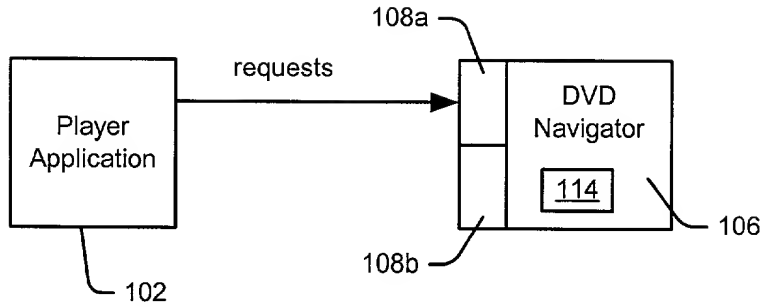
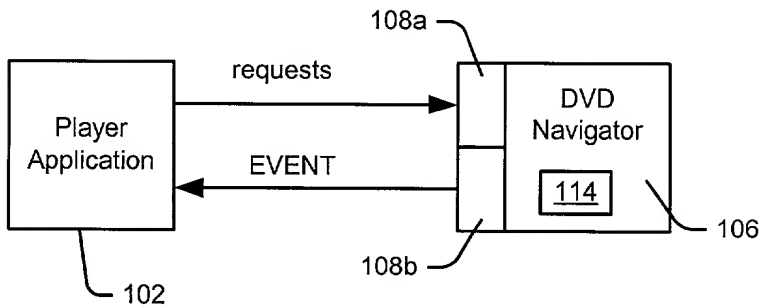


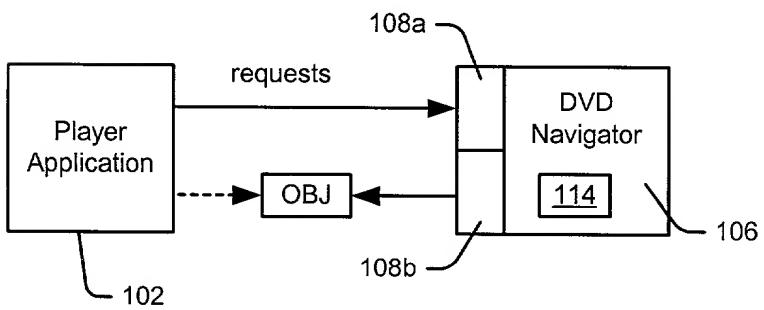
Fig. 2



*Fig. 3*

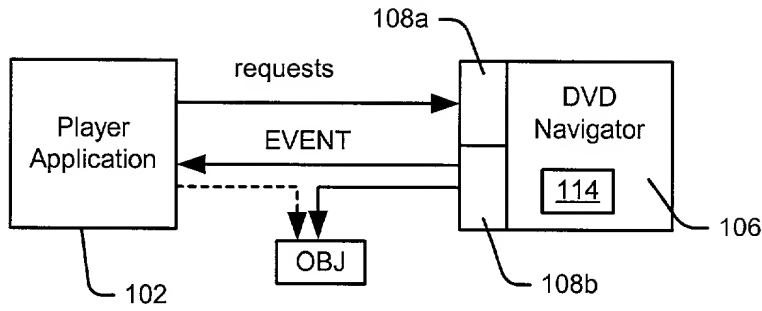


*Fig. 4*

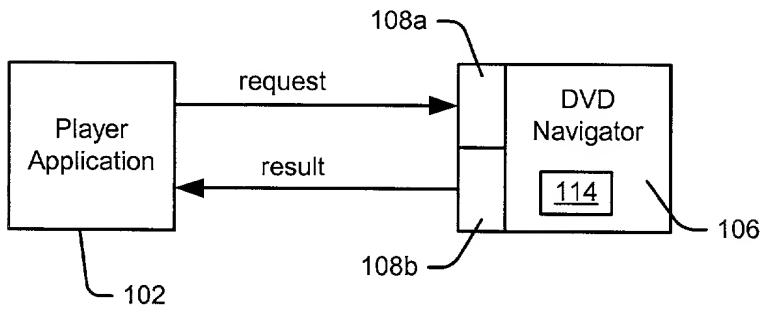


*Fig. 5*

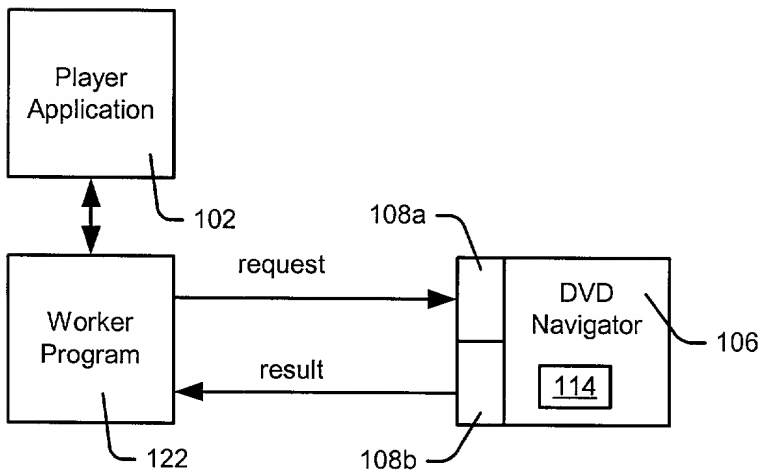
000007-20110226



*Fig. 6*



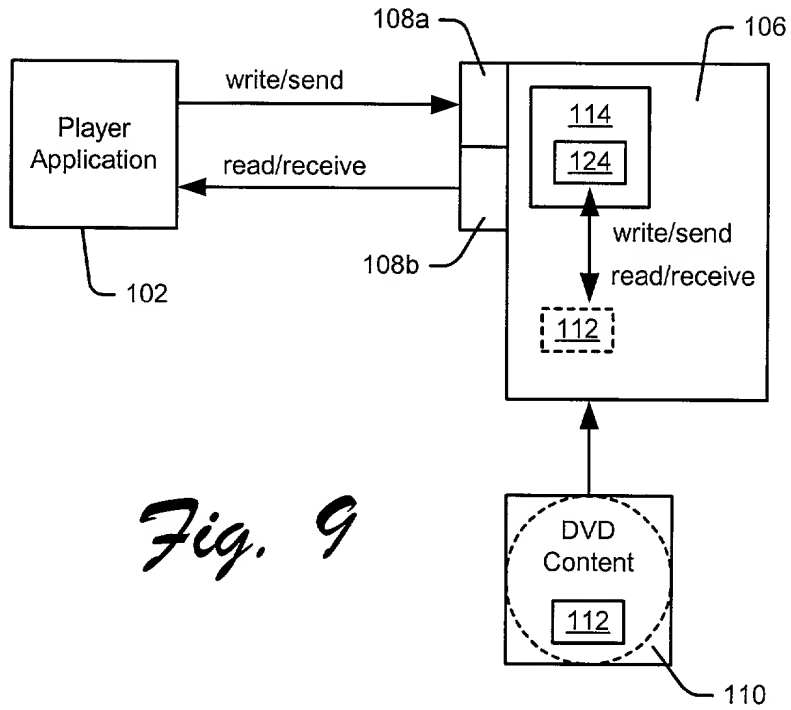
*Fig. 7*



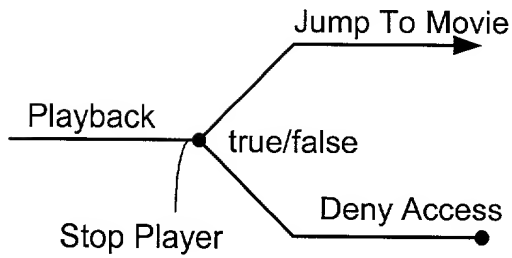
*Fig. 8*

000007-00100000

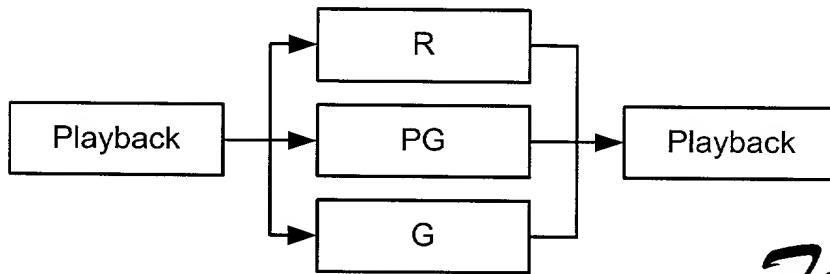




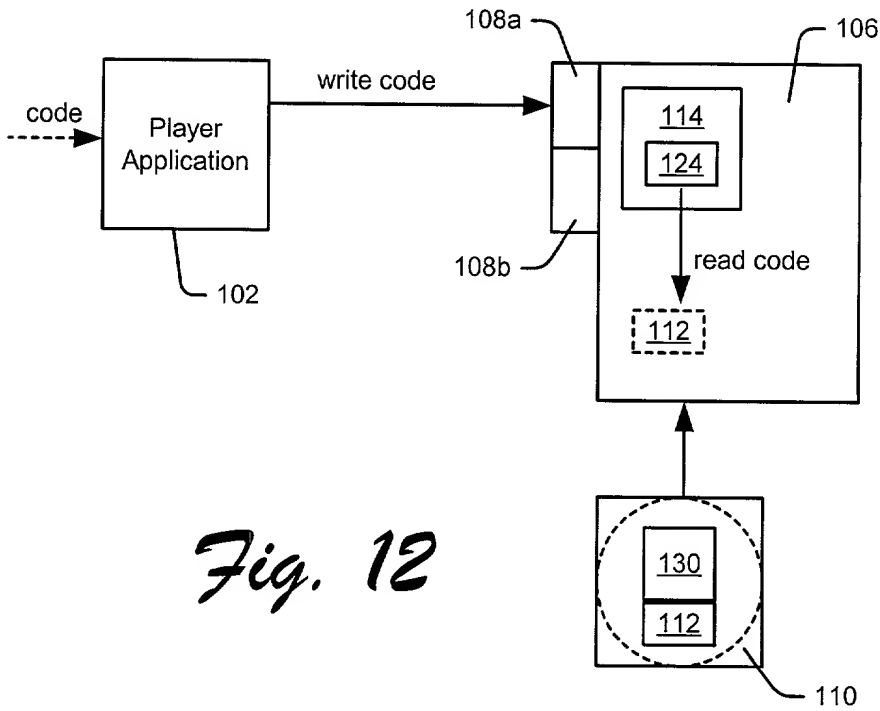
*Fig. 9*



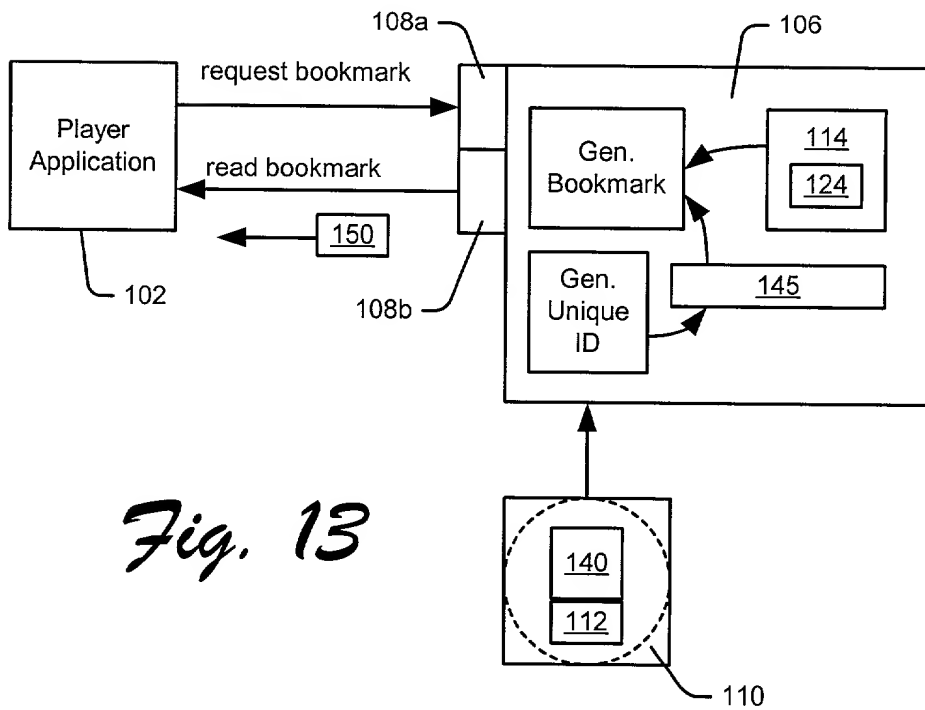
*Fig. 10*



*Fig. 11*

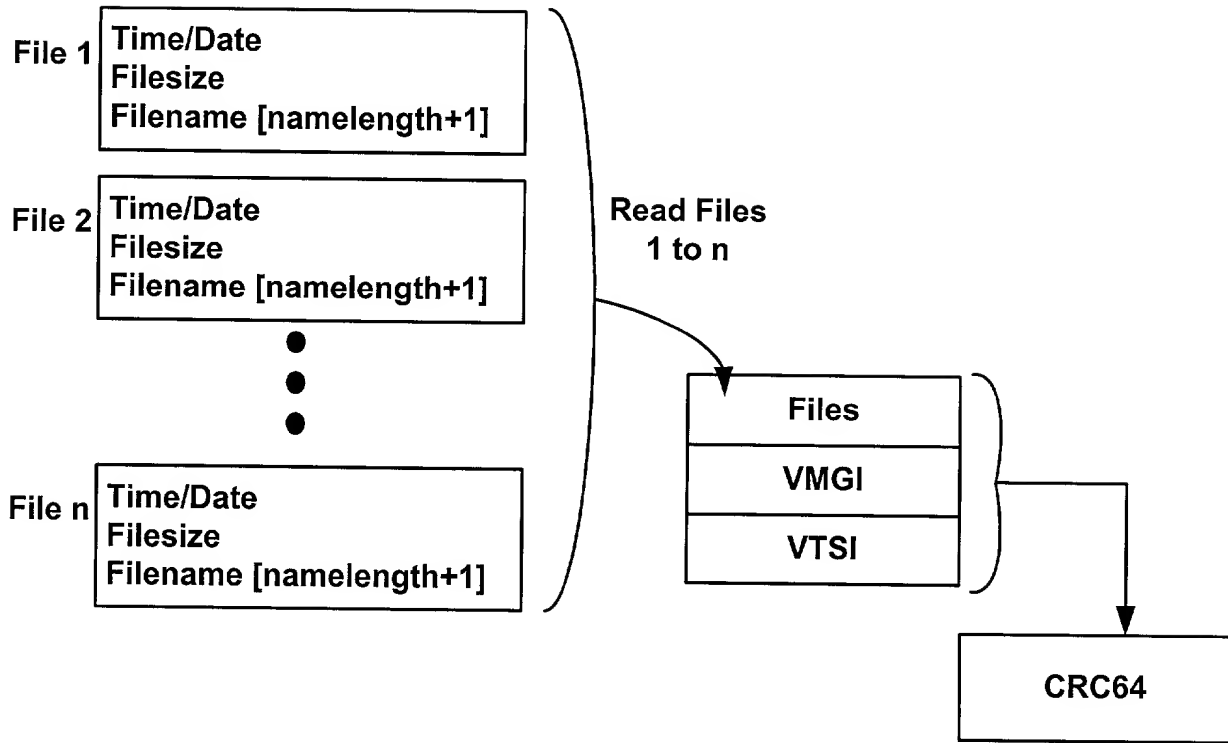


*Fig. 12*



*Fig. 13*

DOC ID: A017260



*Fig. 14*

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Inventorship..... Evans et al.  
Applicant ..... Microsoft Corporation  
Attorney's Docket No. .... MSI-688US  
Title: Improved DVD Navigator and Application Programming Interfaces (APIs)

**DECLARATION FOR PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled "Improved DVD Navigator and Application Programming Interfaces (APIs)," the specification of which is attached hereto.

I have reviewed and understand the content of the above-identified specification, including the claims.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

**PRIOR FOREIGN APPLICATIONS:** no applications for foreign patents or inventor's certificates have been filed prior to the date of execution of this declaration.

**Power of Attorney**

I appoint the following attorneys to prosecute this application and transact all future business in the Patent and Trademark Office connected with this application:  
Lewis C. Lee, Reg. No. 34,656; Daniel L. Hayes, Reg. No. 34,618; Allan T.

002277 2044260

1 Sponseller, Reg. 38,318; Steven R. Sponseller, Reg. No. 39,384; James R.  
 2 Banowsky, Reg. No. 37,773; Lance R. Sadler, Reg. No. 38,605; Michael A. Proksch,  
 3 Reg. No. 43,021; Thomas A. Jolly, Reg. No. 39,241; David A. Morasch, Reg. No.  
 4 42,905; Kasey C. Christie, Reg. No. 40,559; Nathan R. Rieth, Reg. No. 44,302;  
 5 Brian G. Hart, Reg. No. 44,421; Katie E. Sako, Reg. No. 32,628 and Daniel D.  
 6 Crouse, Reg. No. 32,022.

7 Send correspondence to: LEE & HAYES, PLLC, 421 W. Riverside Avenue,  
 8 Suite 500, Spokane, Washington, 99201. Direct telephone calls to: Thomas A. Jolly  
 9 (509) 324-9256.

10  
 11 All statements made herein of my own knowledge are true and that all  
 12 statements made on information and belief are believed to be true; and further that  
 13 these statements were made with the knowledge that willful false statements and the  
 14 like so made are punishable by fine or imprisonment, or both, under Section 1001 of  
 15 Title 18 of the United States Code and that such willful false statement may  
 16 jeopardize the validity of the application or any patent issued therefrom.

17  
 18 \* \* \* \* \*

19 Full name of inventor: Glenn F. Evans  
 20 Inventor's Signature Glenn Evans Date: Nov 22, 2000  
 21 Residence: Kirkland, WA  
 22 Citizenship: Canada  
 23 Post Office Address: 7833 NE 133rd Pl.  
 24 Kirkland, WA 98034  
 25

002211 2044260

\*\*\*\*\*

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

Full name of inventor:

Alok Chakrabarti

Inventor's Signature

*Alok Chakrabarti*

Date: 11.22.2000

Residence:

Bellevue, WA

Citizenship:

India

Post Office Address:

5724 141st Pl. SE  
Bellevue, WA 98006

OFFICE OF THE COMMISSIONER